.NET Full Stack Development Program

Day 8 SQL Server Intro

Outline

- Data Modeling
- ER Diagram
- Normalization
- Database
- SQL
- Constraints

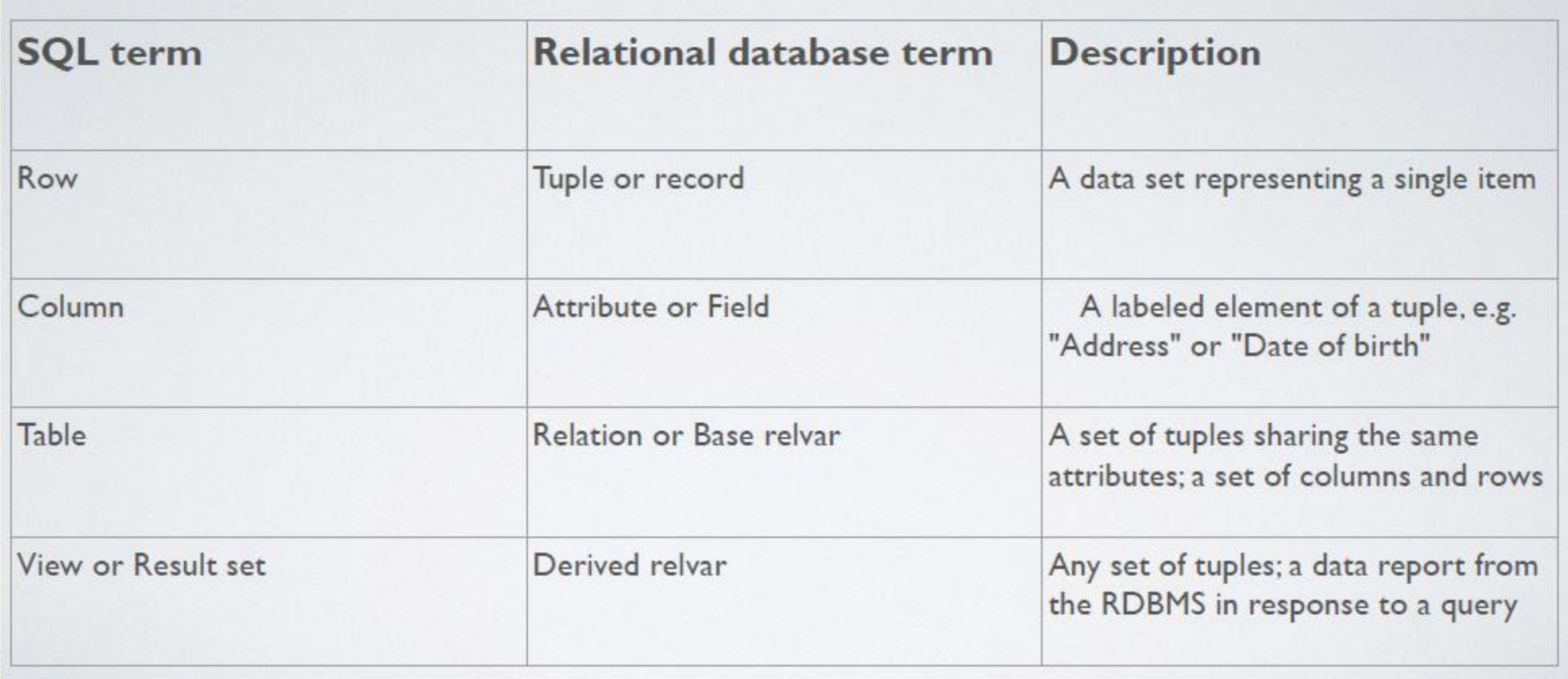
Data Modeling

- Data Model is the process of analyzing **business requirement**, designing and creating physical instance(a plan or a blueprint) for the database or data warehouse.

- It is a stage in SDLC(Software Development Life Cycle)

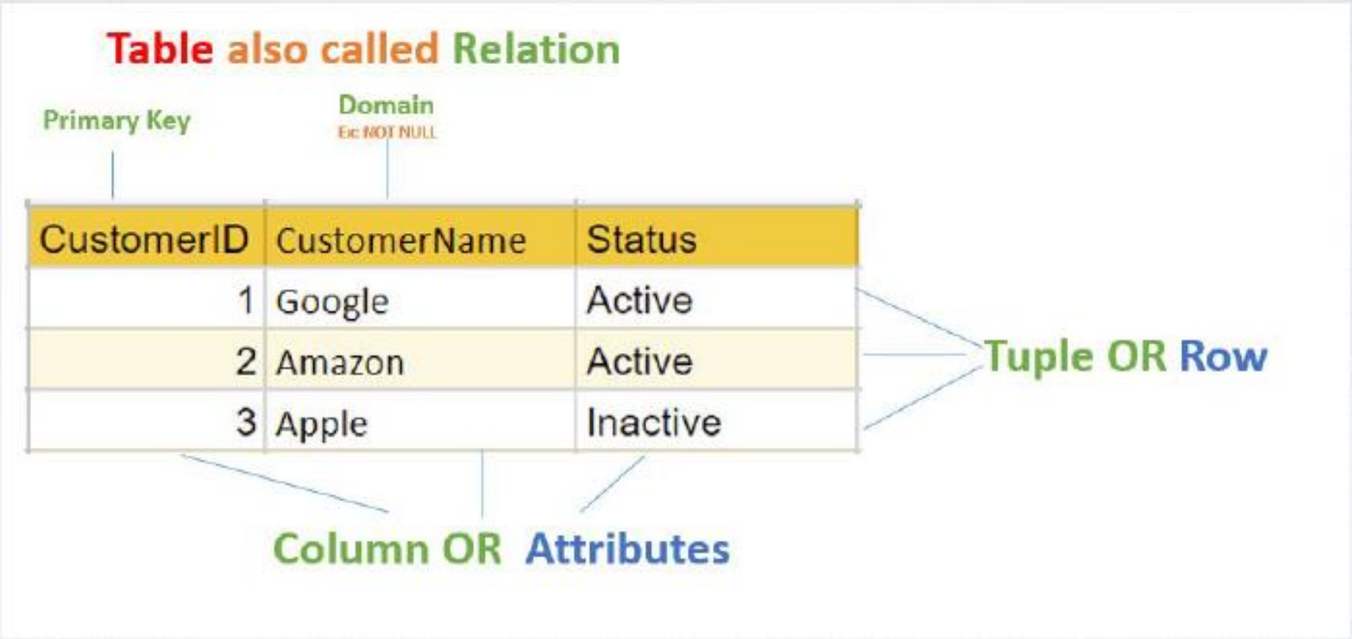
- It also impacts the user interface

Relational Database Terminology



SQL term	Relational database term	Description
Row	Tuple or record	A data set representing a single item
Column	Attribute or Field	A labeled element of a tuple, e.g. "Address" or "Date of birth"
Table	Relation or Base relvar	A set of tuples sharing the same attributes; a set of columns and rows
View or Result set	Derived relvar	Any set of tuples; a data report from the RDBMS in response to a query

Relational Database Terminology

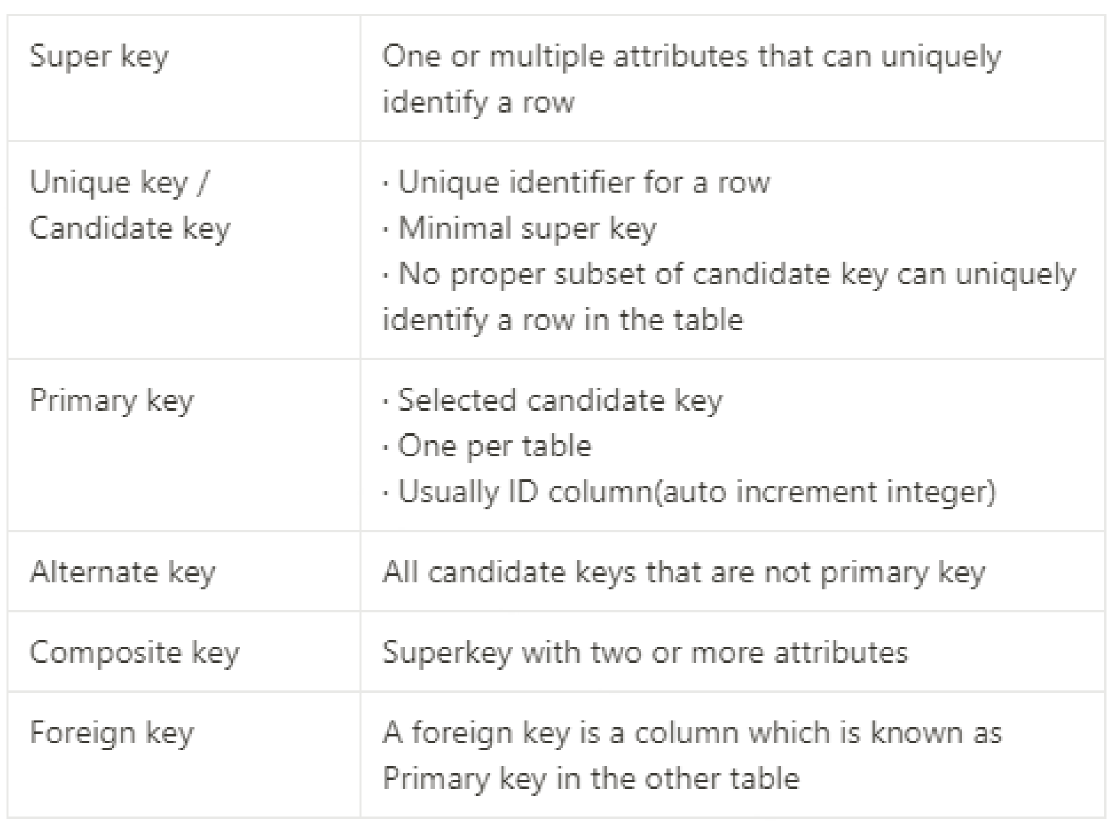


CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Data Modeling Terminology

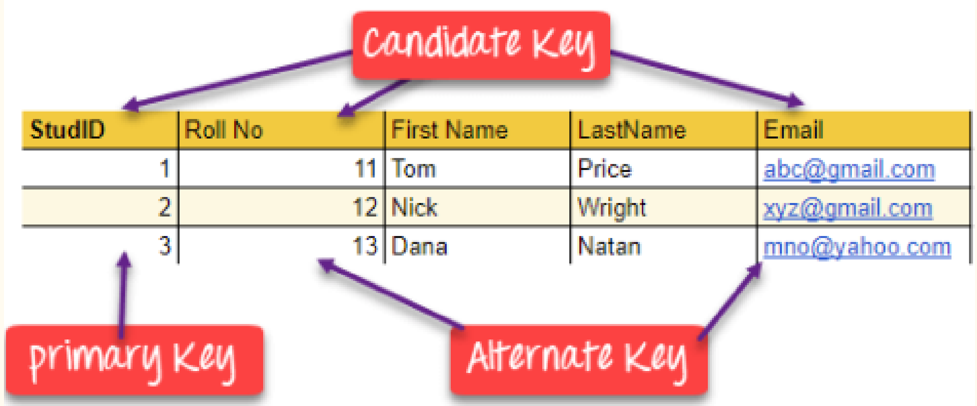
- Entity
  - An item that can exist independently or uniquely identified
- Attribute
  - Column label(name)
- Domain
  - Set of valid values for an attribute
- Relationship
  - How entities relate
- Degree
  - How many entities in a relationship
- Cardinality
  - Measure of participation

Keys Used in Data Modeling



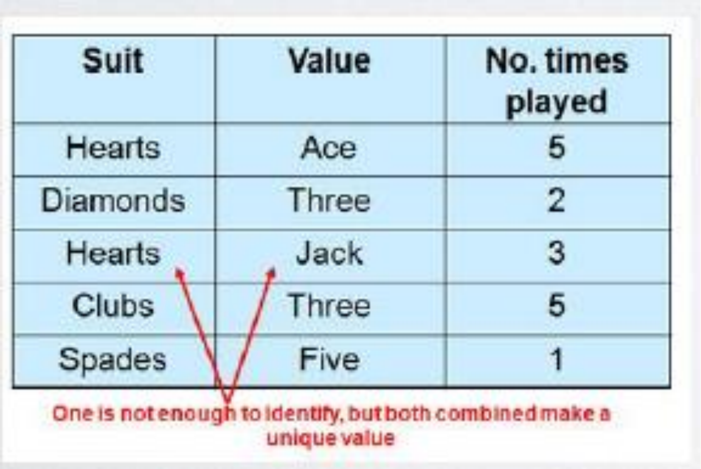
Super key	One or multiple attributes that can uniquely identify a row
Unique key / Candidate key	· Unique identifier for a row · Minimal super key · No proper subset of candidate key can uniquely identify a row in the table
Primary key	· Selected candidate key · One per table · Usually ID column(auto increment integer)
Alternate key	All candidate keys that are not primary key
Composite key	Superkey with two or more attributes
Foreign key	A foreign key is a column which is known as Primary key in the other table

Key Used in Data Modeling



Candidate Key
primary Key
Alternate Key

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com



Suit	Value	No. times played
Hearts	Ace	5
Diamonds	Three	2
Hearts	Jack	3
Clubs	Three	5
Spades	Five	1

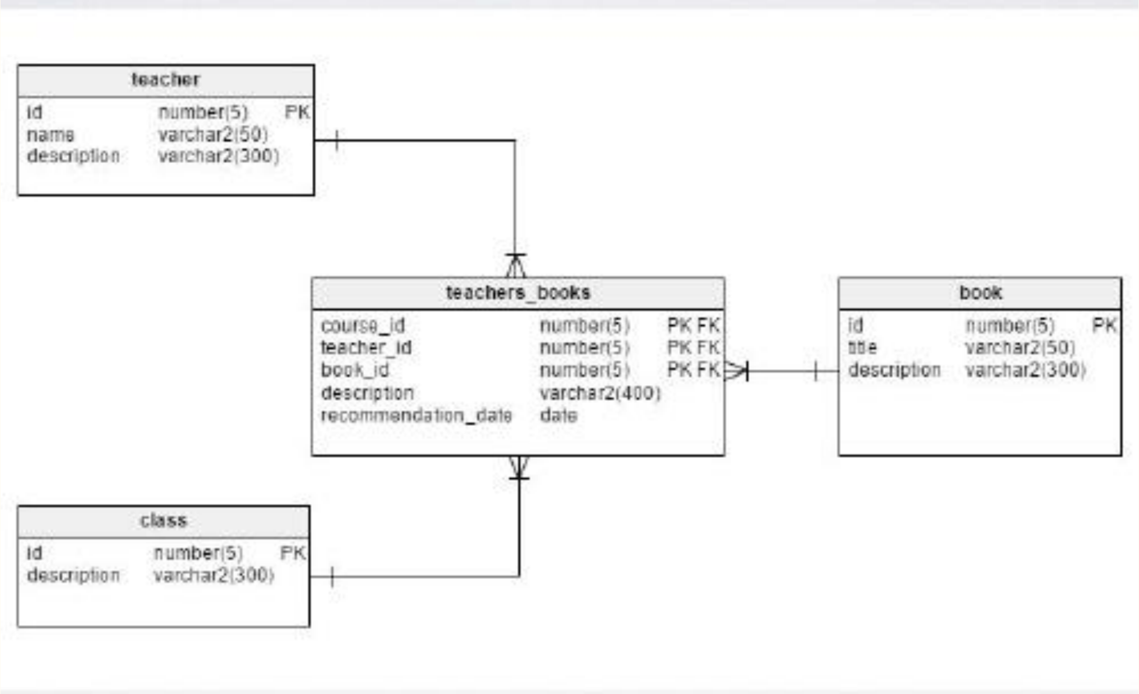
One is not enough to identify, but both combined make a unique value

Cardinality and Degree

- Cardinality is the number of times the entity participates in the relationship
    - One-to-One: One element in entityA may link to one element in entityB and vice versa
    - One-to-Many: One element in entityA may link to many elements in entity but one element in entity may only link to one element in entityA
    - Many-to-One: Reverse A and B in one-to-many
    - Many-to-Many: One element in entityA may link to any number of elements in entity and vice versa
- Degree
    - Degree is the number of entities involved in the relationship and it is usually 2(binary relationship) however Unary and higher degree relationships can exists.

- Cardinality != Degree

Cardinality and Degree



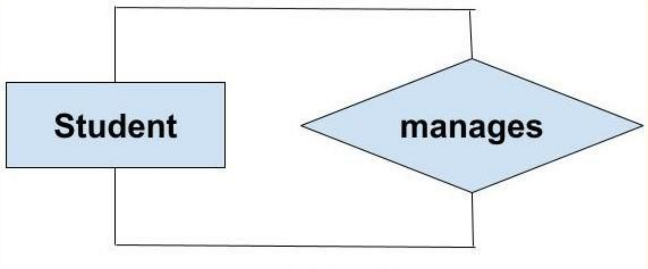
teacher		
id	number(5)	PK
name	varchar2(50)	
description	varchar2(300)	

teachers_books		
course_id	number(5)	PK FK
teacher_id	number(5)	PK FK
book_id	number(5)	PK FK
description	varchar2(400)	
recommendation_date	date	

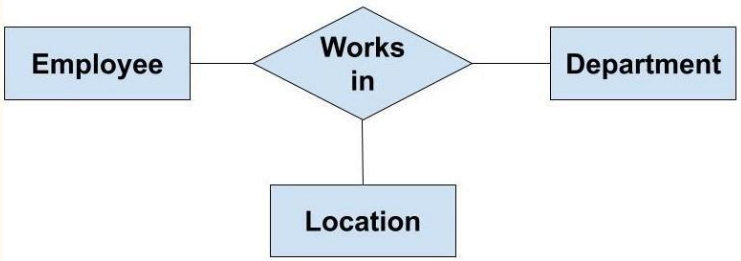
book		
id	number(5)	PK
title	varchar2(50)	
description	varchar2(300)	

class		
id	number(5)	PK
description	varchar2(300)	

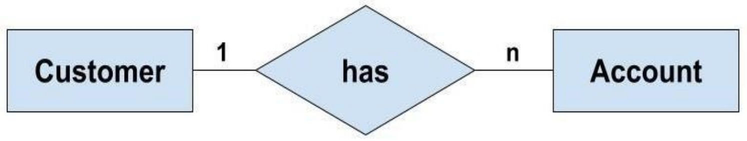
Cardinality and Degree



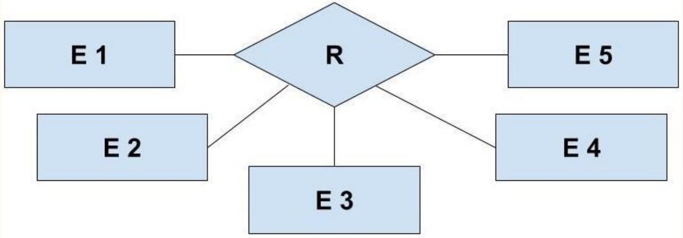
Unary Relationship
(degree 1)



Ternary Relationship
(degree 3)



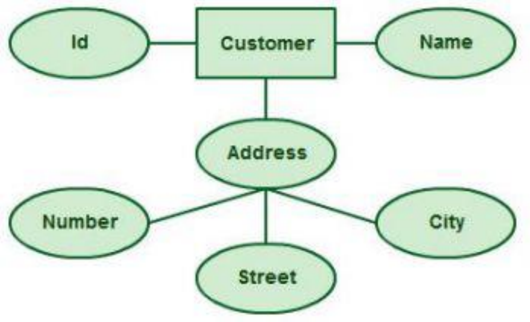
Binary Relationship
(degree 2)



N-ary Relationship

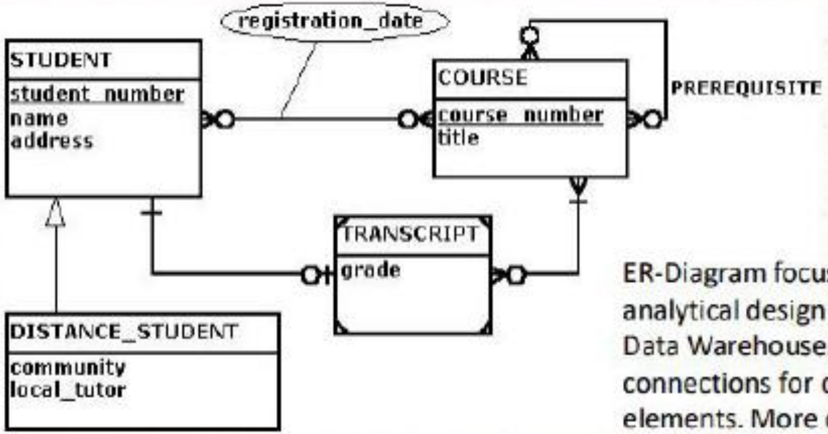
ERD

- ER-Diagram
  - Entity Relationship Diagram
  - Used to create or design a blueprint of the Database or Data Warehouse
  - Design Entities, attributes and show relationships



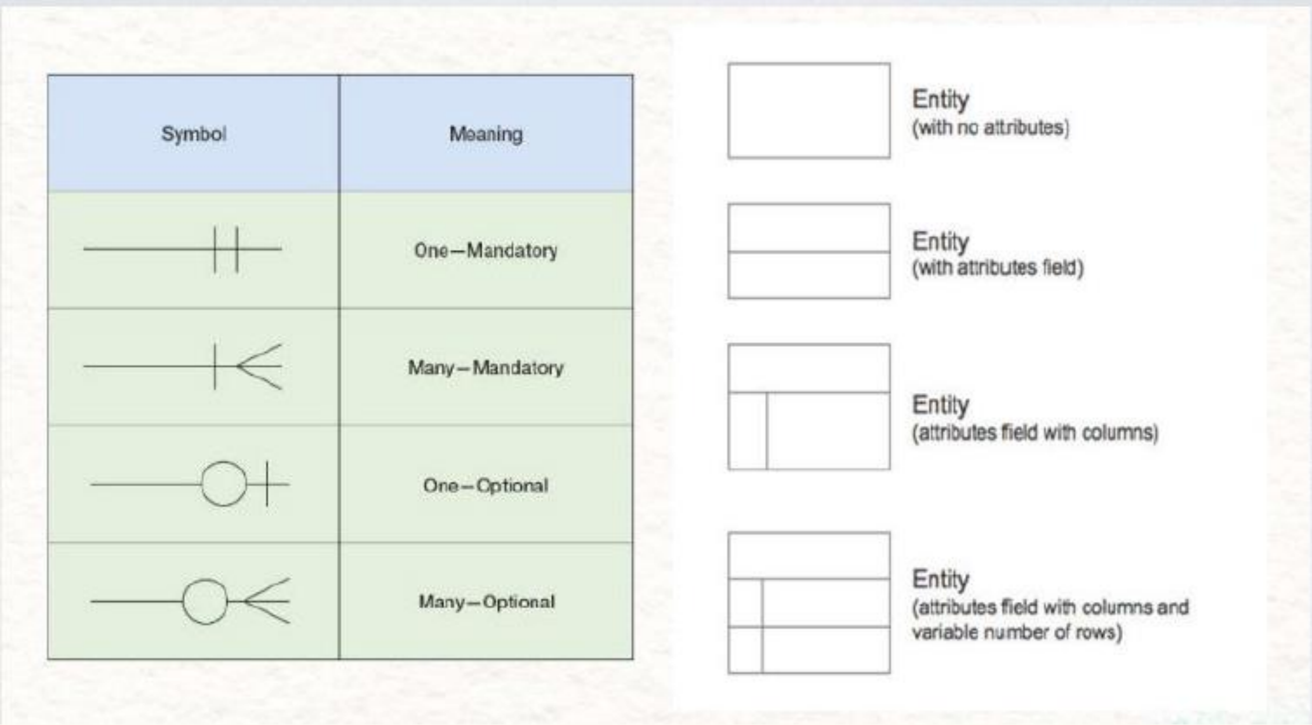
Crow’s Foot Notation

- Connection Symbols display Relationships
- Entity and Attributes in Table like format



ER-Diagram focusing more on analytical design for Database or Data Warehouse. Uses symbols and connections for displaying design elements. More comment used in Physical Data Model.

Crow's Foot Notation



Symbol	Meaning
	One—Mandatory
	Many—Mandatory
	One—Optional
	Many—Optional

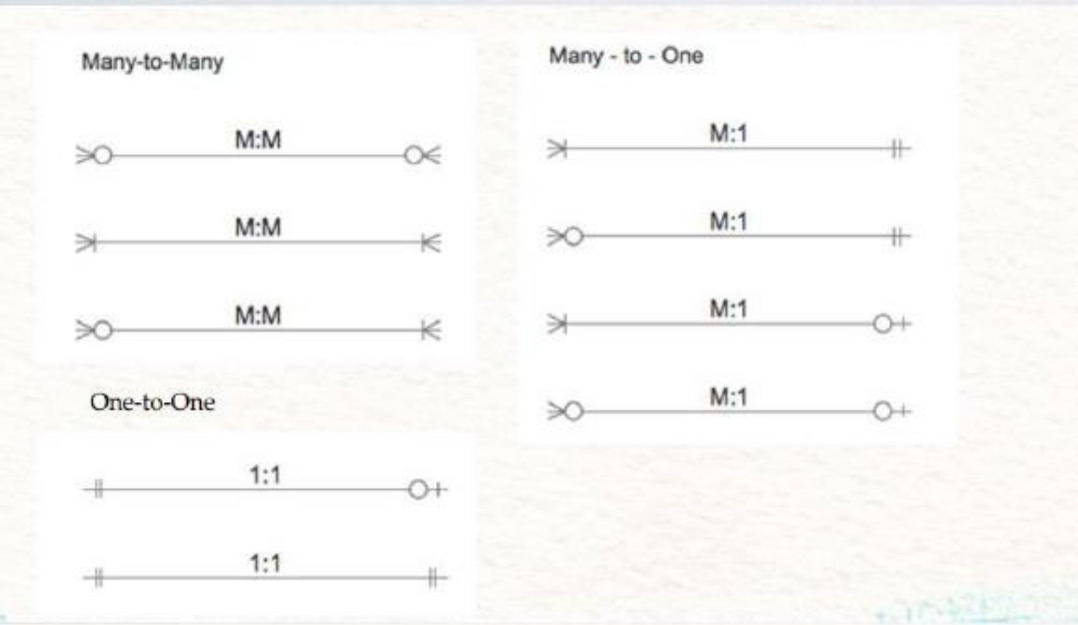
Entity
(with no attributes)

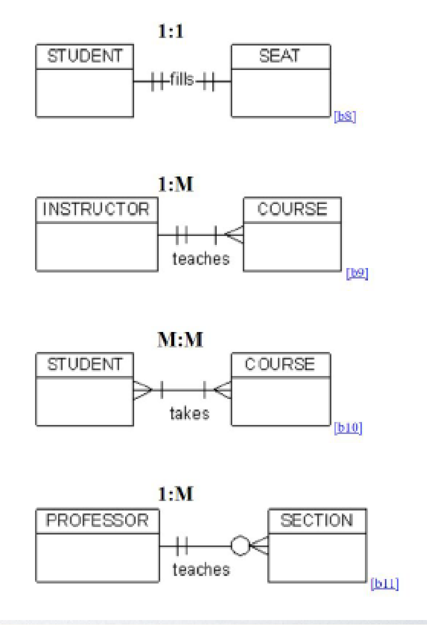
Entity
(with attributes field)

Entity
(attributes field with columns)

Entity
(attributes field with columns and variable number of rows)

Crow’s Foot Notation

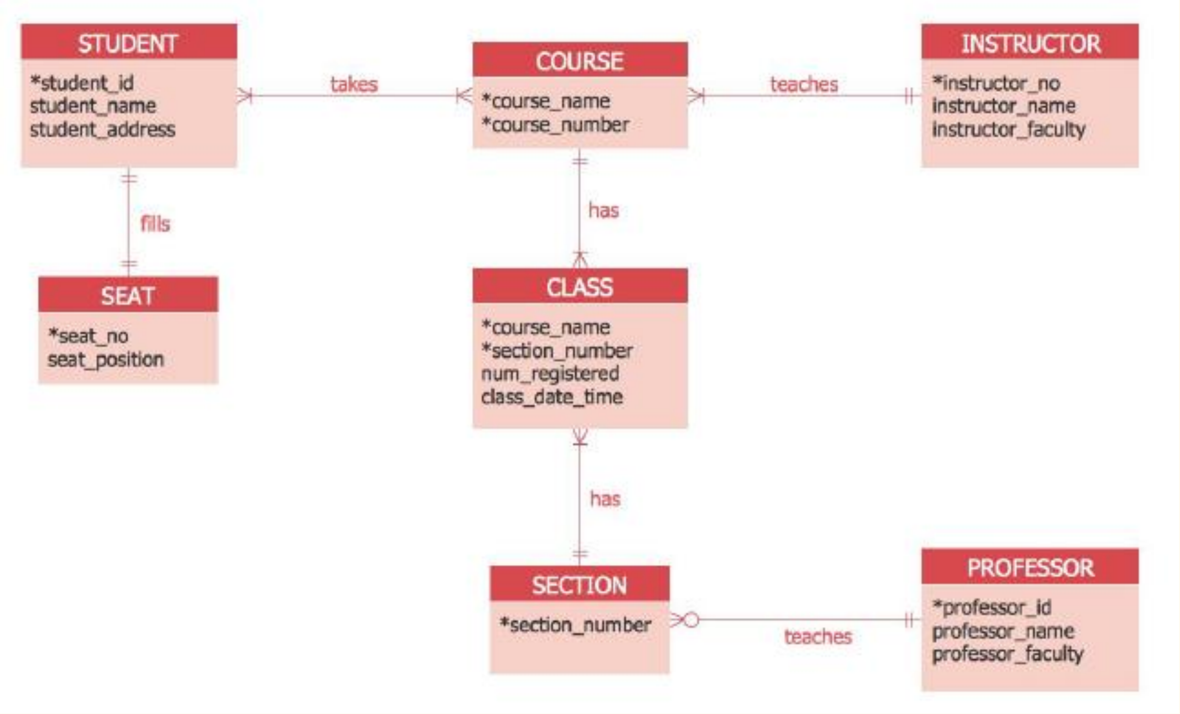




Convert ERD to Tables

- Each entity type becomes a table
- Each single-valued attribute becomes a column
- Derived attributes are ignored/computed column
- Multi-valued attributes are represented by a separate table
- Use Conjunction Table to break up the Many-to-Many relationship
- The key attribute of the entity type becomes the primary key or unique key of the table

Convert ERD to Tables



STUDENT
*student_id
student_name
student_address

COURSE
*course_name
*course_number

INSTRUCTOR
*instructor_no
instructor_name
instructor_faculty

SEAT
*seat_no
seat_position

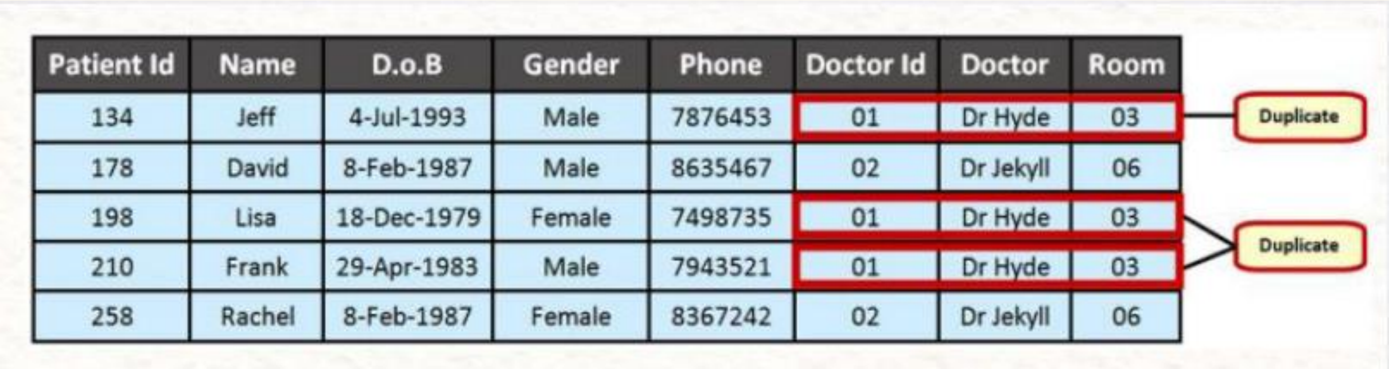
CLASS
*course_name
*section_number
num_registered
class_date_time

SECTION
*section_number

PROFESSOR
*professor_id
professor_name
professor_faculty

Normalization

- Redundancy
  - Values repeated unnecessarily in multiple records or fields within one or more tables



Patient Id	Name	D.o.B	Gender	Phone	Doctor Id	Doctor	Room
134	Jeff	4-Jul-1993	Male	7876453	01	Dr Hyde	03
178	David	8-Feb-1987	Male	8635467	02	Dr Jekyll	06
198	Lisa	18-Dec-1979	Female	7498735	01	Dr Hyde	03
210	Frank	29-Apr-1983	Male	7943521	01	Dr Hyde	03
258	Rachel	8-Feb-1987	Female	8367242	02	Dr Jekyll	06

Duplicate
Duplicate

Normalization

- Anomaly

- When an attempt is made to modify(update, insert into, or delete from) a relation, the following undesirable side-effects may arise in relations that have not been sufficiently normalized

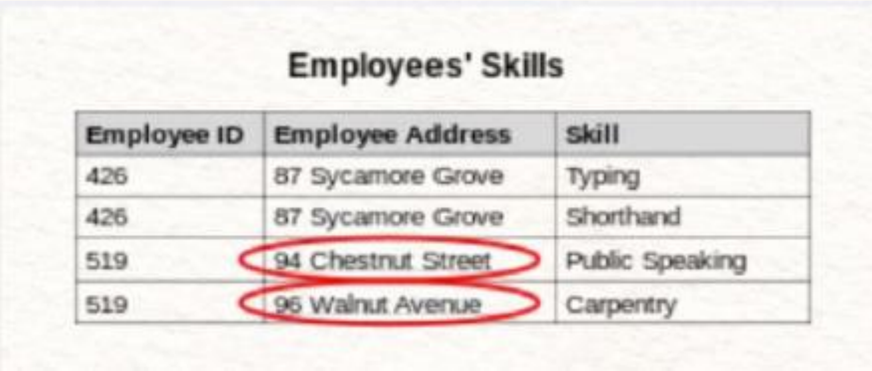
- Anomaly is the issue that may occur because of redundancy

- Types: Update, Insertion and Deletion

Normalization

- Anomaly Update

- The same information can be expressed on multiple rows; Therefore, updates to the relation may result in logical inconsistencies

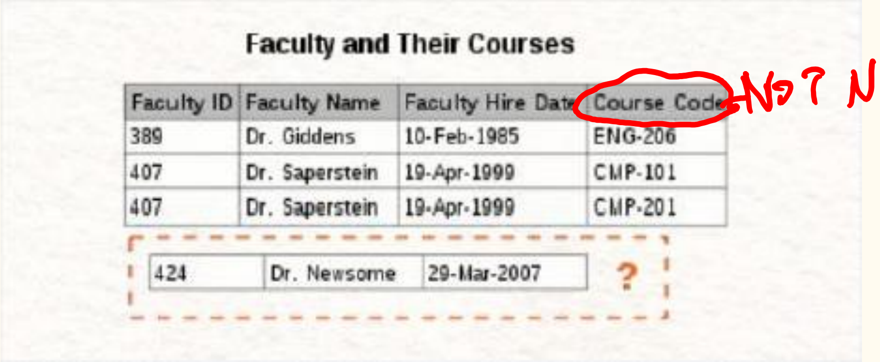


Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Normalization

- Anomaly Insertion
- There are circumstances in which certain facts cannot be recorded at all



Faculty and Their Courses

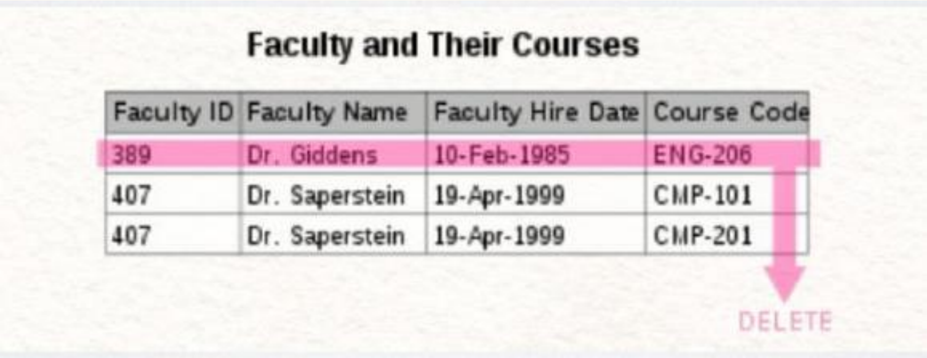
Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

NOT NULL

Normalization

- Anomaly Deletion

- Under certain circumstances, deletion of data representing certain facts necessitates deletion of data representing completely different facts



Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

DELETE

Normalization

- Main goal of normalization

- Reduce redundancy, avoid anomaly and create a well-structured series of table without error or inconsistencies

- Minimize redesign when extending the database structure
    - New type of data can be accommodated without changing existing structure too much

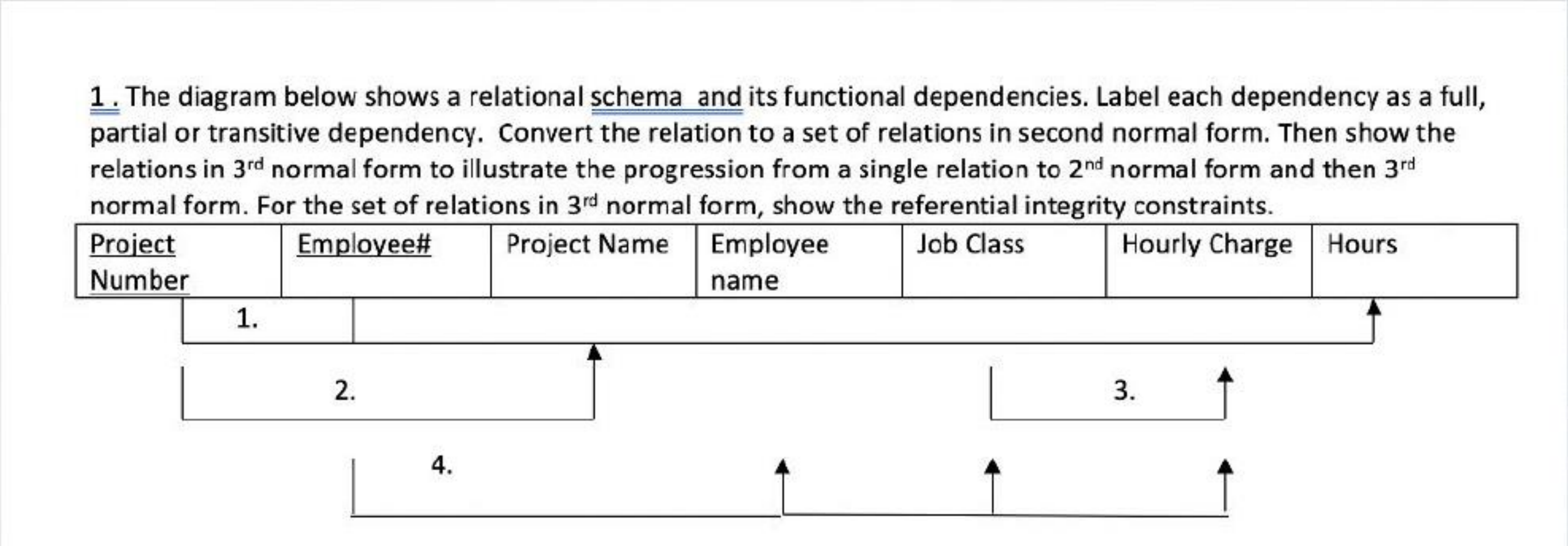
- Ensure data dependencies are properly enforced by data integrity constraints(Entity Integrity, Referential Integrity, Domain Integrity)

Normalization

- Functional Dependencies
  - Functional Dependencies are how different attributes relate in a table
  - At this level, we focus on individual tables
  - We see how individual attributes relate to the keys in the table
    - Primary Key & Candidate Keys = Prime Attributes
    - Attributes that aren't keys = Non-Prime Attributes

- Types of Dependencies
  - Full Dependencies – Depends on all prime attributes fully
  - Partial Dependencies – Depends on some Prime Attributes
  - Transitive Dependencies – Depends on an attribute that depends on a Prime Attribute

Normalization



1 . The diagram below shows a relational schema and its functional dependencies. Label each dependency as a full, partial or transitive dependency. Convert the relation to a set of relations in second normal form. Then show the relations in 3rd normal form to illustrate the progression from a single relation to 2nd normal form and then 3rd normal form. For the set of relations in 3rd normal form, show the referential integrity constraints.

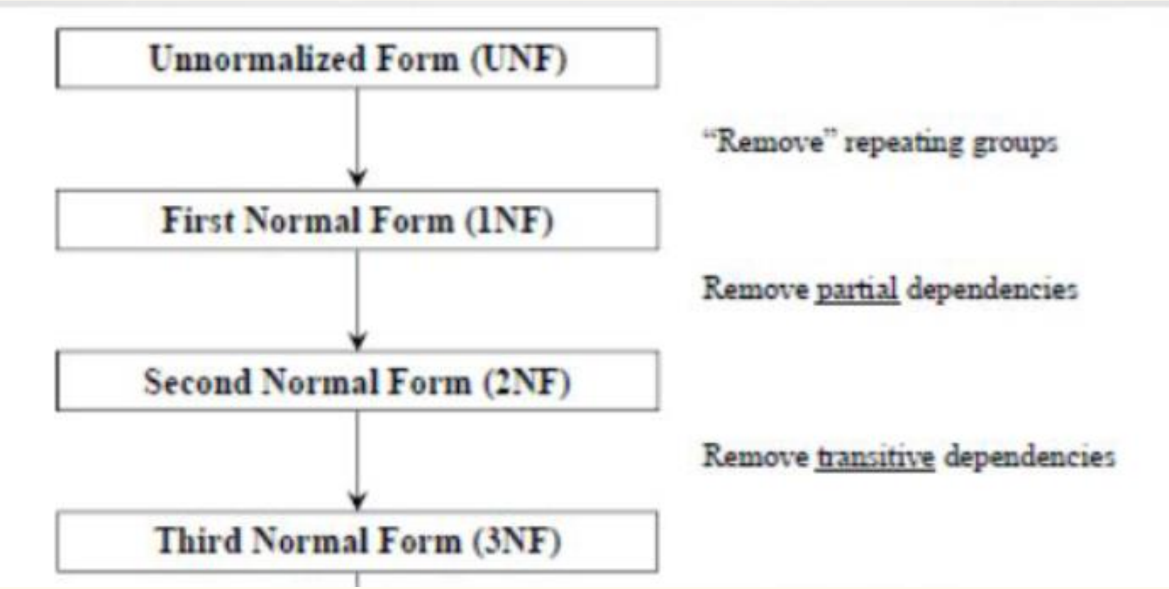
Project Number	Employee#	Project Name	Employee name	Job Class	Hourly Charge	Hours

Normalization

- First Normal Form
  - Each table cell should contain a single value
  - Each record needs to be unique
- Second Normal Form
  - Meets all of 1NF
  - Makes sure all non-prime attributes are fully dependent on a prime attribute
- Third Normal Form
  - Meets 1NF and 2NF
  - Every non-prime attribute is non-transitively dependent on the prime attributes

Normalization

- Process



Unnormalized Form (UNF)

“Remove” repeating groups

First Normal Form (1NF)

Remove partial dependencies

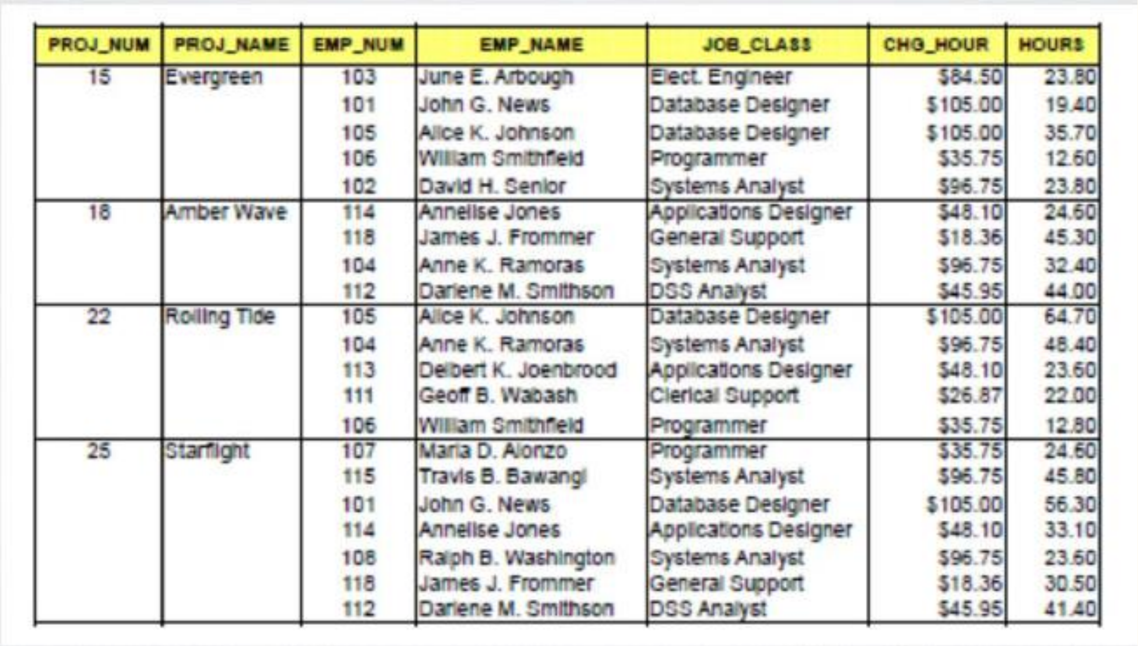
Second Normal Form (2NF)

Remove transitive dependencies

Third Normal Form (3NF)

Normalization

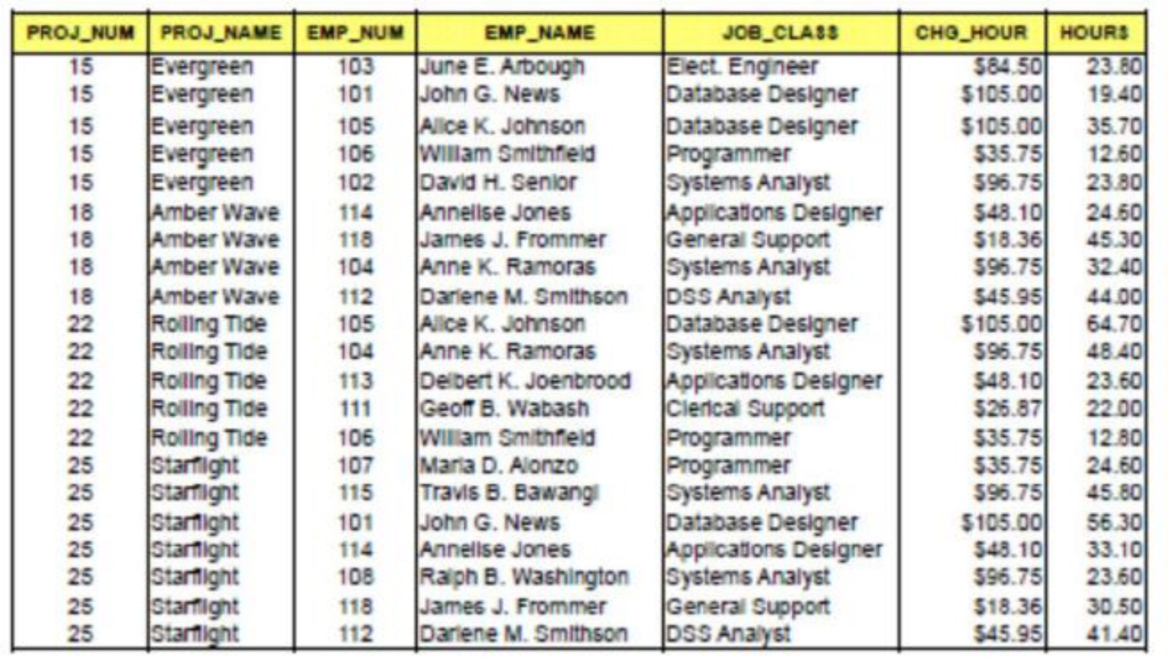
- UNF



PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.80
		101	John G. News	Database Designer	$105.00	19.40
		105	Alice K. Johnson	Database Designer	$105.00	35.70
		106	William Smithfield	Programmer	$35.75	12.60
		102	David H. Senior	Systems Analyst	$96.75	23.80
18	Amber Wave	114	Annelise Jones	Applications Designer	$48.10	24.60
		118	James J. Frommer	General Support	$18.36	45.30
		104	Anne K. Ramoras	Systems Analyst	$96.75	32.40
		112	Darlene M. Smithson	DSS Analyst	$45.95	44.00
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.70
		104	Anne K. Ramoras	Systems Analyst	$96.75	48.40
		113	Delbert K. Joenbrood	Applications Designer	$48.10	23.60
		111	Geoff B. Wabash	Clerical Support	$26.87	22.00
		106	William Smithfield	Programmer	$35.75	12.80
25	Starflight	107	Maria D. Alonzo	Programmer	$35.75	24.60
		115	Travis B. Bawangi	Systems Analyst	$96.75	45.80
		101	John G. News	Database Designer	$105.00	56.30
		114	Annelise Jones	Applications Designer	$48.10	33.10
		108	Ralph B. Washington	Systems Analyst	$96.75	23.60
		118	James J. Frommer	General Support	$18.36	30.50
		112	Darlene M. Smithson	DSS Analyst	$45.95	41.40

Normalization

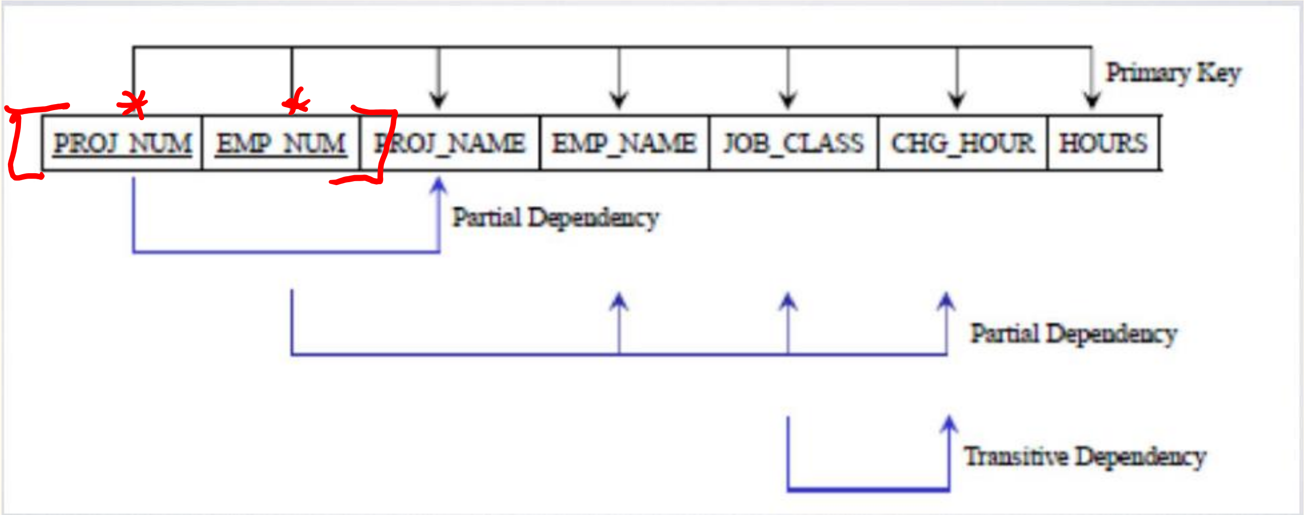
- After 1NF



PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.80
15	Evergreen	101	John G. News	Database Designer	$105.00	19.40
15	Evergreen	105	Alice K. Johnson	Database Designer	$105.00	35.70
15	Evergreen	106	William Smithfield	Programmer	$35.75	12.60
15	Evergreen	102	David H. Senior	Systems Analyst	$96.75	23.80
18	Amber Wave	114	Annelise Jones	Applications Designer	$48.10	24.60
18	Amber Wave	118	James J. Frommer	General Support	$18.36	45.30
18	Amber Wave	104	Anne K. Ramoras	Systems Analyst	$96.75	32.40
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	$45.95	44.00
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.70
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	$96.75	48.40
22	Rolling Tide	113	Delbert K. Joenbrood	Applications Designer	$48.10	23.60
22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	$26.87	22.00
22	Rolling Tide	106	William Smithfield	Programmer	$35.75	12.80
25	Starflight	107	Maria D. Alonzo	Programmer	$35.75	24.60
25	Starflight	115	Travis B. Bawangi	Systems Analyst	$96.75	45.80
25	Starflight	101	John G. News	Database Designer	$105.00	56.30
25	Starflight	114	Annelise Jones	Applications Designer	$48.10	33.10
25	Starflight	108	Ralph B. Washington	Systems Analyst	$96.75	23.60
25	Starflight	118	James J. Frommer	General Support	$18.36	30.50
25	Starflight	112	Darlene M. Smithson	DSS Analyst	$45.95	41.40

Normalization

- Dependencies



PROJ_NUM	EMP_NUM	PROJ_NAME	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS

Primary Key

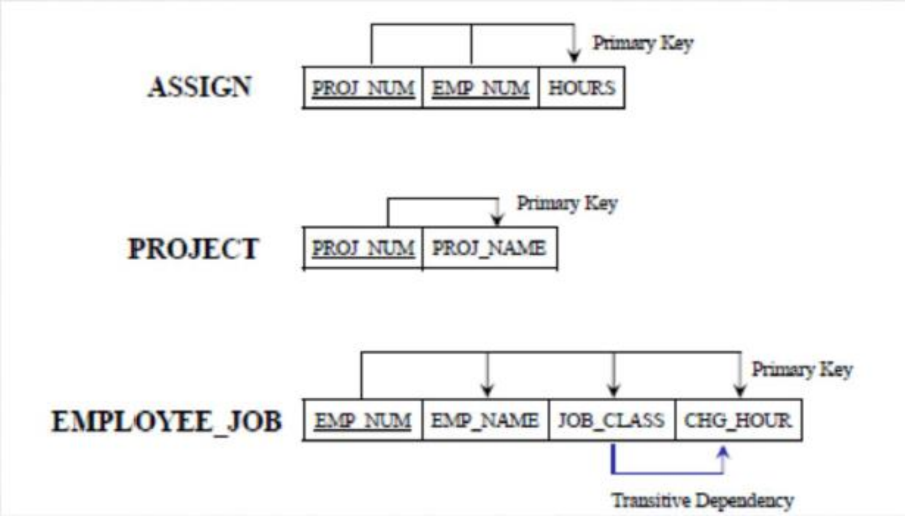
Partial Dependency

Partial Dependency

Transitive Dependency

Normalization

- After 2NF(Remove partial dependencies)



Primary Key

ASSIGN	PROJ_NUM	EMP_NUM	HOURS

Primary Key

PROJECT	PROJ_NUM	PROJ_NAME

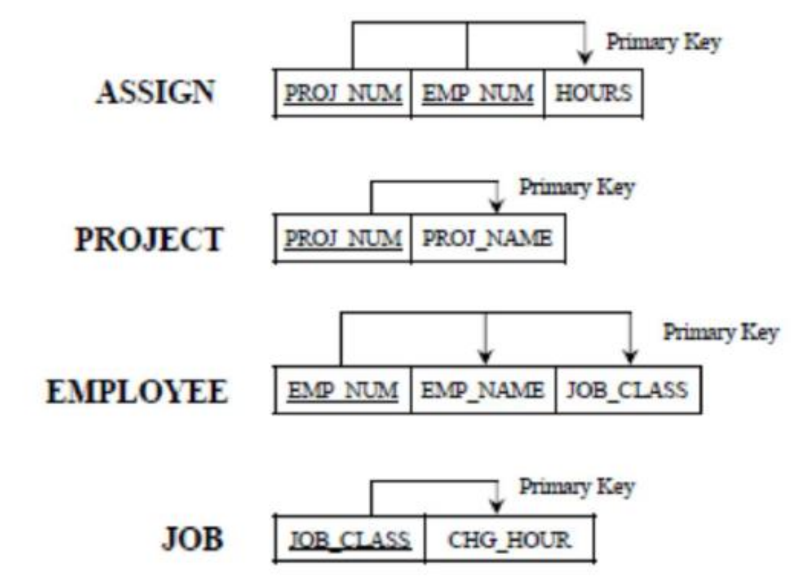
Primary Key

EMPLOYEE_JOB	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR

Transitive Dependency

Normalization

- After 3NF(Remove transitive dependency)



ASSIGN: PROJ_NUM | EMP_NUM | HOURS

Primary Key

PROJECT: PROJ_NUM | PROJ_NAME

Primary Key

EMPLOYEE: EMP_NUM | EMP_NAME | JOB_CLASS

Primary Key

JOB: JOB_CLASS | CHG_HOUR

Primary Key

Normalization

- When do we normalize?
  - Usually normalize in database with lots of read/writes
  - Not too much fetching/only need to fetch a small subset of the data
  - Data integrity is crucial especially because there us a lot of user input

- Cons
  - Normalization is an expensive process
  - Designing can be difficult – Good for final design, not testing
  - More tables leads to more time
  - Joins are costly, having more tables can cause slowing

Database Integrities

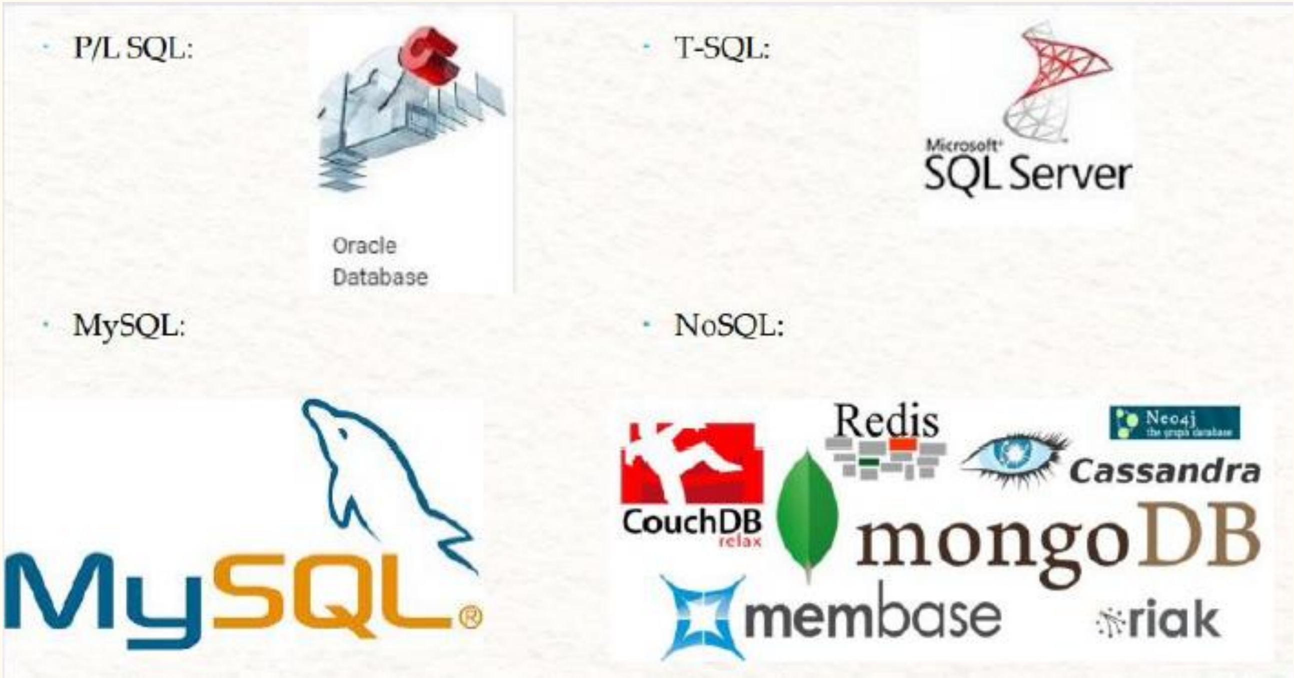
- Entity Integrity
  - Design of the table or entity
  - String PK with no nulls or repeats

- User-Defined Integrity
  - Rules or constraints applied by the user to maintain rules of design

- Domain Integrity
  - Correct and proper domains specified with proper use of columns

- Referential Integrity
  - Proper FK setup with proper PK reference
  - Good design for connection and joins

Database Management System



- P/L SQL:
- T-SQL:
- MySQL:
- NoSQL:

Oracle
Database
Microsoft
SQL Server
MySQL
Redis
Neo4j
the graph database
Cassandra
CouchDB
relax
mongoDB
membase
riak

Relational Database vs. Non-Relational Database

- Relational Database(SQL)
  - Traditional way of storing data
  - Data is stored in tables with rows and columns
  - Rigid schema with well-defined relationships
  - Difficult to scale
  - Examples: SQL Server, Oracle, MySQL

- Non-Relational Database(NoSQL)
  - Developed more recently
  - Data can be stored in a variety of formats(JSON documents, key-value pairs, wide-column, graphs)
  - Flexible schema with loose relationships
  - Easily scalable
  - Examples: MongoDB, Redis

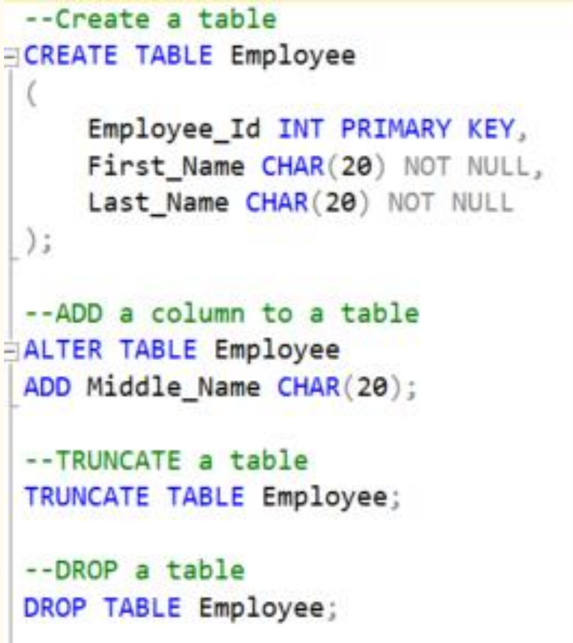
SQL Introduction

SQL(Structured Query Language)

- Data Definition Language(DDL)
- (DML)
- (DCL)
- (DQL)

SQL

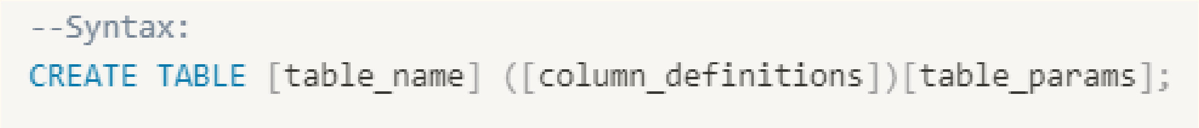
- DDL – Data Definition Language
- Create
- Alter
- Drop
- Truncate



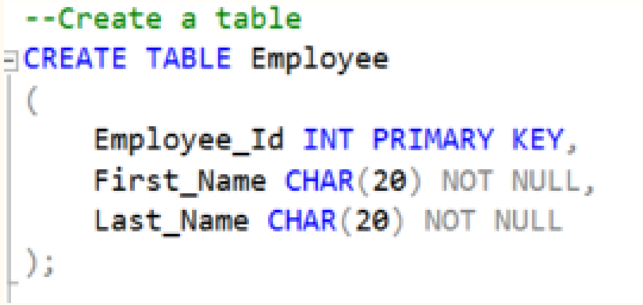
```
--Create a table
CREATE TABLE Employee
(
    Employee_Id INT PRIMARY KEY,
    First_Name CHAR(20) NOT NULL,
    Last_Name CHAR(20) NOT NULL
);

--ADD a column to a table
ALTER TABLE Employee
ADD Middle_Name CHAR(20);

--TRUNCATE a table
TRUNCATE TABLE Employee;

--DROP a table
DROP TABLE Employee;
```

Create

- This command builds a new table and has a predefined syntax.



```
--Syntax:
CREATE TABLE [table_name] ([column_definitions])[table_params];
```

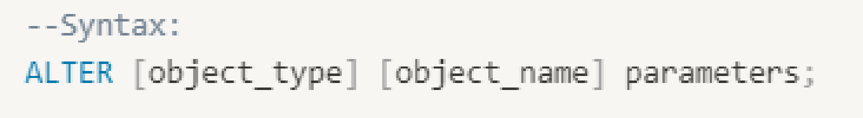


```
--Create a table
CREATE TABLE Employee
(
    Employee_Id INT PRIMARY KEY,
    First_Name CHAR(20) NOT NULL,
    Last_Name CHAR(20) NOT NULL
);
```

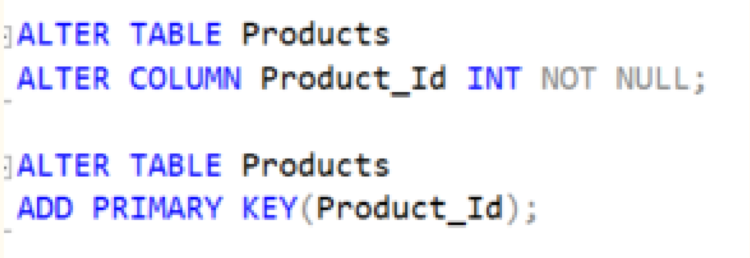
- In this example, the string CHAR is used to specify the data type. Other data types can be DATE, INT, DECIMAL etc.

Alter

- An Alter command modifies an existing database table. This command can add up additional column, drop existing columns and even change the data type of columns involved in a database table.



```
--Syntax:
ALTER [object_type] [object_name] parameters;
```

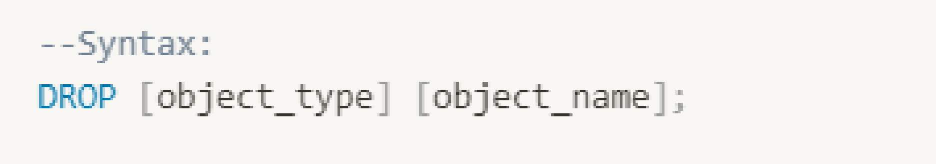


```
ALTER TABLE Products
ALTER COLUMN Product_Id INT NOT NULL;

ALTER TABLE Products
ADD PRIMARY KEY(Product_Id);
```

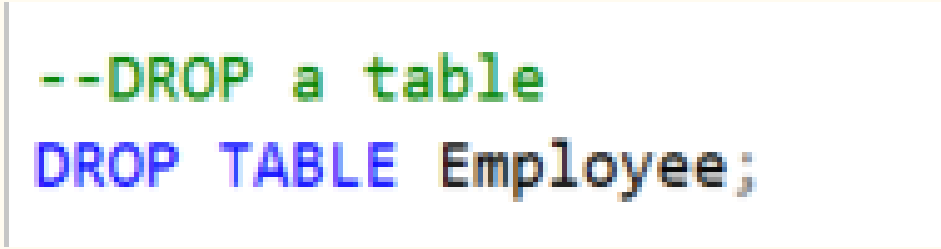
- In this example, we added a unique primary key to the table to add a constraint and enforce a unique value. The constraint “Product_Id” is a primary key and is on the Products table.

Drop

- A drop command is used to delete objects such as a table, index or view. A DROP statement **cannot be rolled back**, so once an object is destroyed, there’s no way to recover it.



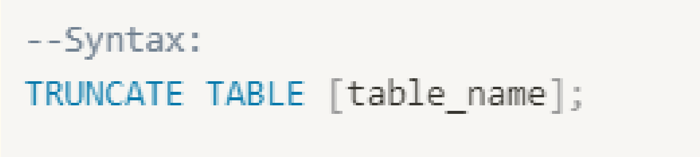
```
--Syntax:
DROP [object_type] [object_name];
```



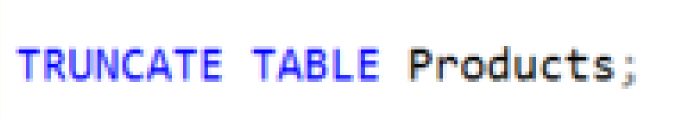
```
--DROP a table
DROP TABLE Employee;
```

Truncate

- Similar to DROP, the TRUNCATE statement is used to quickly remove all records from a table. However, unlike DROP that completely destroys a table, TRUNCATE preserves it’s full structure to be reused later.



```
--Syntax:
TRUNCATE TABLE [table_name];
```



```
TRUNCATE TABLE Products;
```

SQL

- DML – Data Manipulation Language
- Insert
- Update
- Delete

DML

- Insert Statements
  - Insert statements are used to input data into tables
  - The order of data specified should match order of columns
  - If you don’t know the order of columns, specify each column name in the insert statement
- For example
  - Inserting data matching order
    - Insert into TableName values(1, ‘Name’),(2, ‘Name’);
  - Inserting data without knowing matching order
    - Insert into TableName(Col2, Col1) values(‘Name’, 1),(‘Name’, 2);

DML

- Update Statements
  - Update statements are used to change or modify data inside a table
  - Specify single row depending on statement
  - Use unique identifiers to get correct rows

- For Example
  - Update using unique column
    - Update TableName Set Name = ‘Tim’ Where ID = 2;
  - Update using non-unique column
    - Update TableName Set Name = ‘Hal’ Where Name = ‘Bob’

DML

- Delete Statements
  - Delete statements are used to remove specific rows inside a table
  - Use unique values to identify the correct rows to delete
  - Delete leave logs and continue identity values
- For Example
  - Deleting rows using specific unique values
    - Delete from TableName Where ID = 1;
  - Deleting rows using non-unique values
    - Delete From TableName Where Name = ‘Jim’

SQL

- DCL – Data Control Language

- DCL is syntax used to control what permission a user can have

- You can create Roles that users can be grouped into a specific permission there

- You can choose what tables someone in a role can access and even what statements can be performed

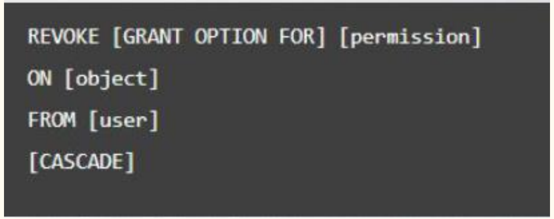
DCL

- Grant
  - Used to “grant” or give permissions to a user or role

- Example:
  - Create table permission to a role
    - Grant Create Table to TestRole
  - Add user to the role
    - Exec sp_addrolemember ‘TestRole’, ‘TestUser’

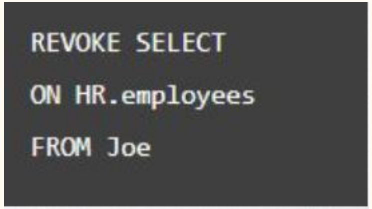
DCL

- Revoke
  - Revoke is used to take away permissions given, whether they are Grant or Deny permissions.



```
REVOKE [GRANT OPTION FOR] [permission]
ON [object]
FROM [user]
[CASCADE]
```

- Example
  - Revoking grant permission
  - Revoke Create Table to TestRole

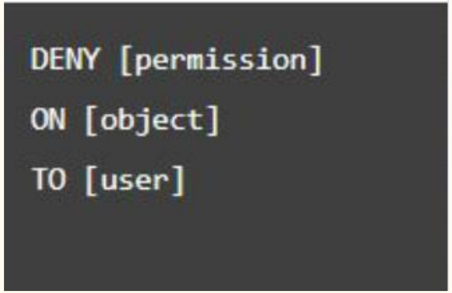


```
REVOKE SELECT
ON HR.employees
FROM Joe
```

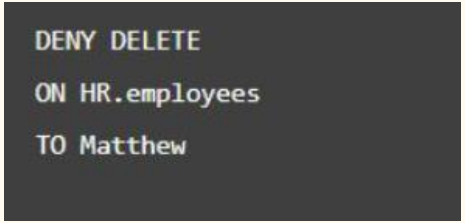
DCL

- Deny
- Deny is preventing someone from having access at all.
- Deny is not the same as revoke, as revoke is made to take back, deny is made to say NO
- If grant and deny permission are given to the same role, deny will take over

- Example
  - Deny Create Table to TestRole



```
DENY [permission]
ON [object]
TO [user]
```

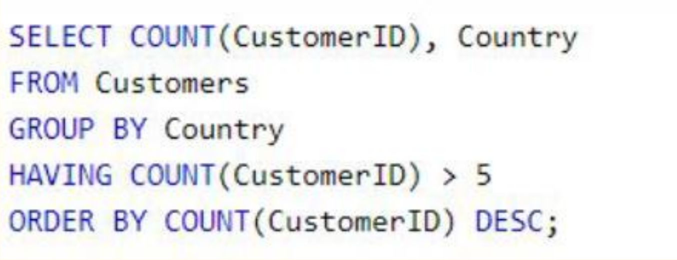


```
DENY DELETE
ON HR.employees
TO Matthew
```

SQL

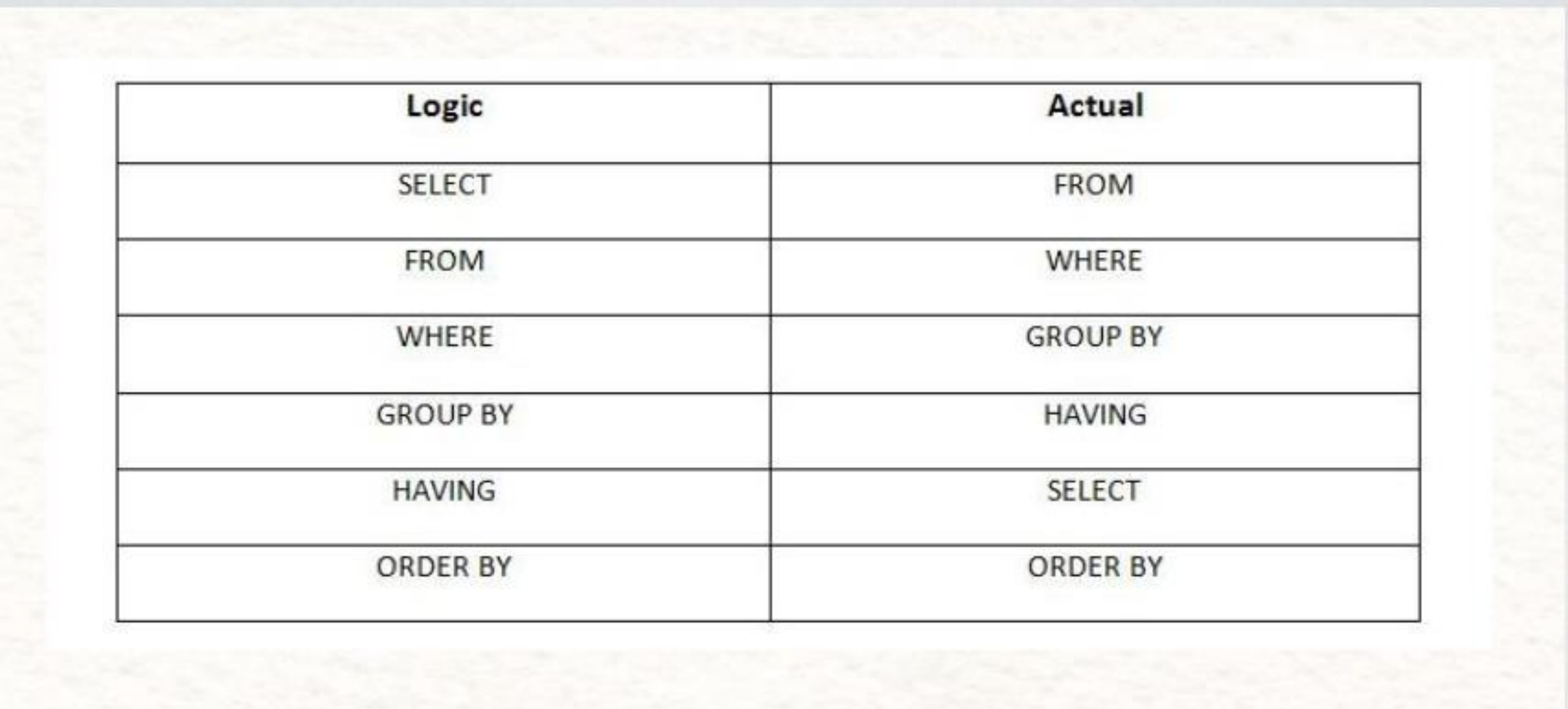
- DQL – Data Query Language
- Select From Where
    - Select: Pick up which column of data you’d like to fetch
    - From: Select which table or data set to fetch.
    - Where: Specific a criteria to sort data by use operators(Filter).
        - Operators: In, Or, And

- Group by, Having, Order by
    - Group by – Used to combine similar values in column
    - Having – filter conditions for aggregate only
    - Order by – display the data by order by a specific column



```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

DQL Execution Order



Logic	Actual
SELECT	FROM
FROM	WHERE
WHERE	GROUP BY
GROUP BY	HAVING
HAVING	SELECT
ORDER BY	ORDER BY

Constraint

- A constraint is usually associated with a table and is created with a CREATE CONSTRAINT SQL statement.

- They define certain properties that data in a database must comply with

Constraint

- Key Constraints
  - Primary Key
    - 1 per table
    - Unique Clustered index
    - Not Null
  - Unique Key
    - 999 per table
    - Unique Non-Clustered index
    - 1 Null Allowed
  - Foreign Key
    - Cannot exist before PK
    - Must be deleted before PK

- Other Constraints
  - Null, Not Null
    - Are nulls allowed
  - Check
    - Data must meet rule
  - Default
    - If nothing, then this
  - Data types
    - Char(2) –States (NY, CA…)
    - Varchar(10) – Names…
    - Money -- Money

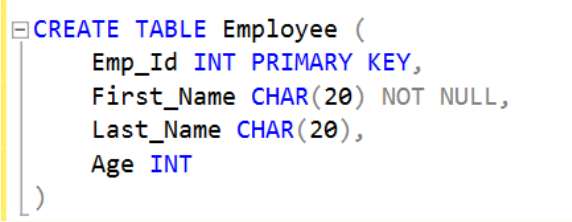
Constraint

- Adding constraints to a table
    - Constraints: used to specify rules for the data in table
    - Type: NOT NULL, UNIQUE, PRIMARY KE

- How?
    - Create table then alter to add constraints
    - Add constraint as we specify the column in a table
    - Add constraint in the same create statement, after we specified the table

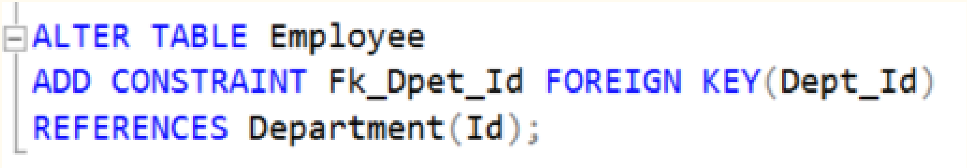
Constraint

- Add constraint as we specify the column in a table



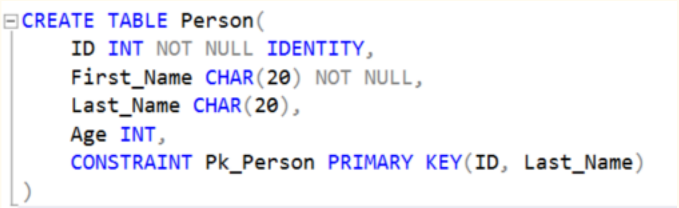
```
CREATE TABLE Employee (
    Emp_Id INT PRIMARY KEY,
    First_Name CHAR(20) NOT NULL,
    Last_Name CHAR(20),
    Age INT
)
```

- Add constraint in the same create statement after we specified the table



```
ALTER TABLE Employee
ADD CONSTRAINT Fk_Dpet_Id FOREIGN KEY(Dept_Id)
REFERENCES Department(Id);
```

- Create a table then alter it to add constraints



```
CREATE TABLE Person(
    ID INT NOT NULL IDENTITY,
    First_Name CHAR(20) NOT NULL,
    Last_Name CHAR(20),
    Age INT,
    CONSTRAINT Pk_Person PRIMARY KEY(ID, Last_Name)
)
```

Questions?